Finite Buffers and Fast Multicast

Peter B. Danzig

Computer Science Division
University of California, Berkeley
Berkeley, California 94720

Abstract. *When many or all of the recipients of a multicast message respond to the multicast's sender, their responses may overflow the sender's available buffer space. Buffer overflow is a serious, known problem of broadcast-based protocols, and can be troublesome when as few as three or four recipients respond. We develop analytical models that calculate the expected number of buffer overflows that can be used to estimate the number of buffers necessary for an application. The common cure for buffer overflow requires that recipients delay their responses by some random amount of time in order to increase the minimum spacing between response messages, eliminate collisions on the network, and decrease the peak processing demand at the sender. In our table driven algorithm, the sender tries to minimize the multicast's latency, the elapsed time between its initial transmission of the multicast and its reception of the final response, given the number of times (rounds) it is willing to retransmit the multicast. It includes in the multicast the time interval over which it anticipates receiving the response, the round timeout. We demonstrate that the latency of single round multicasts exceeds the latency of multiple round multicasts. We show how recipients minimize the sender's buffer overflows by independently choosing their response times as a function of the round's timeout, sender's buffer size, and the number of other recipients.*

1. Introduction

A multicast message is a message simultaneously sent (broadcast) to a group of recipients. When a recipient site receives a multicast transmission, it formulates and forwards its response to the multicast's original sender. These numerous, closely spaced, responses may overwhelm the multicast's sender, causing buffer overflow in its network interface, in its operating system, or in its outside-the-kernel communication protocol processes. If the operating system implements protocol processing in the user's address space [1], responses passing through the operating system may overflow buffers in the user process that initiated the multicast because this is swapped out, blocked, or does not receive adequate processor time. We illustrate these overflow points in Figure 1.

Since real systems have finite memory, they have finite buffers. However most finite buffer queueing analyses apply only to stationary, homogeneous arrival processes; little work exists in the analysis of finite buffer systems [2] [3] [4] with non-stationary, heterogeneous arrival processes. Standard blocking system analyses from text books do not apply to this problem because they deal with stationary Poisson and stationary general arrival processes. The arrival process of responses to a multicast is neither stationary nor homogeneous. It is not homogeneous since the responses come from both fast and slow machines; it is not stationary since the arrival rate decreases as responses are received. In this paper, we calculate buffer overflow losses due to responses from multicast transmissions. Having calculated the number and distribution of losses, we describe an algorithm that minimizes the time to receive all responses to a multicast given that the sender is willing to retransmit a certain number of times. In the remainder of this section we define terminology, present several situations for which multicast is appropriate, and describe our model of the network.

We say the *sender* sends a *multicast* to the *recipients* and that each recipient *responds* with its *response*. The elapsed time

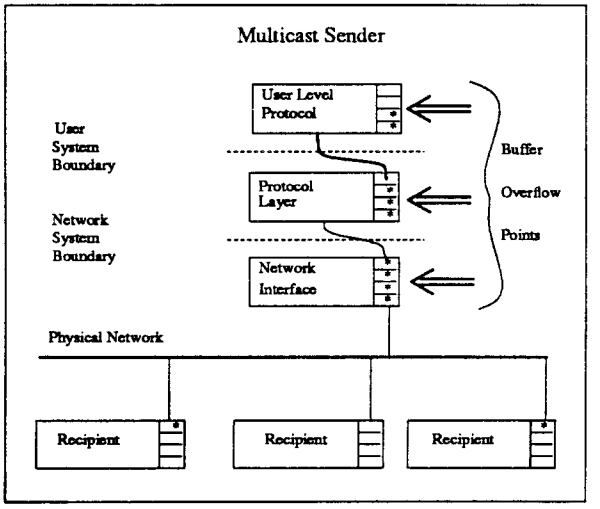

Figure 1. Where losses may occur.

This work has been supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by the Naval Electronic Systems Command under Contract No. N00039-84-C-0089 and by an American Electronics Association Faculty Development Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring organizations or of the U.S. Government.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1988		**00-00-1988 to 00-00-1988**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Finite Buffers and Fast Multicast	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

When many or all of the recipients of a multicast message respond to the multicast's sender, their responses may overflow the sender's available buffer space. Buffer overflow is a serious, known problem of broadcast-based protocols, and can be troublesome when as few as three or four recipients respond. We develop analytical models that calculate the expected number of buffer overflows that can be used to estimate the number of buffers necessary for an application. The common cure for buffer overflow requires that recipients delay their responses by some random amount of time in order to increase the minimum spacing between response messages, eliminate collisions on the network, and decrease the peak processing demand at the sender. In our table driven algorithm, the sender tries to minimize the multicast's latency, the elapsed time between its initial transmission of the multicast and its reception of the final response, given the number of times (rounds) it is willing to retransmit the multicast. It includes in the multicast the time interval over which it anticipates receiving the response, the round timeout. We demonstrate that the latency of single round multicasts exceeds the latency of multiple round multicasts. We show how recipients minimize the sender's buffer overflows by independently choosing their response times as a function of the round's timeout, sender's buffer size, and the number of other recipients.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**10**	

between the instant the multicast is sent and the instant a recipient's response is received is the recipient's *response time*. The probability distribution function of the recipient's response time is the *response time distribution*. The probability distribution function of the time devoted by the sender to process a response and free the buffer in which it is stored is the sender's *service time distribution*. Additional time that a recipient holds its already calculated response before sending it to the sender is the recipient's *backoff time*. The number of times the sender rebroadcasts the message is the number of multicast *rounds*. The time window associated with each round during which the sender collects responses is the round's *timeout*, see Figure 2. We treat responses that arrive after the round's timeout expires as losses during that round and assume their response time is resampled for each subsequent round. The elapsed time between the sender's initial transmission of the multicast and its receipt of the last responses is the multicast *latency*. We do not measure time in seconds or in milliseconds, but rather treat it as a unitless quantity. We require, however, that the service time and response time distributions be expressed in identical unit systems.

1.1. Our Model of the Network

The analyses in the following sections assume (1) the network is unreliable, and some sites may not hear transmissions received at other sites; (2) except where noted, the network is otherwise idle; (3) messages do not collide on the wire; and (4) except where noted, messages require no transmission time. Equivalently, assumptions 3 and 4 say the network bandwidth is infinite. These assumptions make our analysis possible, and tend to overestimate the number of buffer overflows experienced in practice for the following reasons. Competing network traffic gives the sender extra time to service its buffers. Ignoring message length permits responses to arrive closer together than physically permitted by the network. Ignoring collisions and their prescribed rescheduling reduces the time period over which the responses arrive. Since collisions do not occur on ring networks, these analyses perhaps better model rings than CSMA/CD networks. Finally (5), we assume that all recipients of a multicast respond to the sender if only to acknowledge that they correctly received the message.

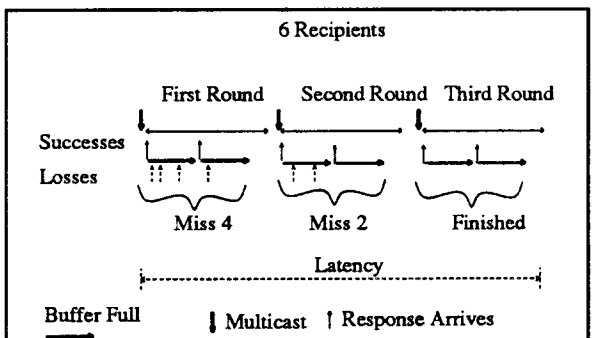

Figure 2. Three rounds of multicast.

1.2. Why Overflow is Important

Speed mismatches at the interface of system layers cause buffer overflow. The bit rate of fiber optic networks exceeds the rate at which data can be copied into most computer's memory and definitely exceeds the rate at which the computer's protocol processing runs. These speed mismatches will always exist. Although various proponents of broadcast-based protocols believe that the problem of buffer overflows is solved in practice, this is not true. Let us review their arguments.

Although many existing network interfaces can not keep pace with the network and possess few buffers, some argue that interface technology is improving and buffer overflow at the interface will not be a problem in the near future. While true for the slow Ethernet, we argue that as interface technology advances, network speeds are advancing many times faster, and these problems will reappear. FDDI, a 100 MBit/Sec ring network, exceeds the the memory access speed of most existing computers. Since a multicast's responses come from many computers and outstrips the rate at which a single computer can send messages, these responses will have to be buffered in the interface. The cost of high speed buffer memory will limit the buffer size within these interfaces, and buffer overflow at the interface will again be a problem. Regardless of network speed, the various computers on a network will always have disparate speeds and buffering capacity. Fast computers must communicate with slow computers. Historicly, interface technology lags behind the introduction of new networks.

Some argue that the number of acknowledgments can be reduced. The ordered broadcast algorithms proposed by Chang and Maxemchuk [5] eliminate most acknowledgements because recipients do not respond to the sender. The number of acknowledgements is a tunable parameter. However, their algorithms suffer long delivery latency times, are complex, and do not apply to situations where the recipients must respond by transmitting data to the sender.

Others argue that acknowledgments can be missed. The V kernel deterministically missed two out of four and one out of three responses [6], Cheriton says this is not a problem as he needs only the first response, but not everyone needs only the first, or the first few responses.

We must understand the fundamental statistics behind buffer overflow to address the problem adequately, wherever it occurs. Buffer overflow can occur within the operating system and protocol processing processes. For example, the Berkeley 4.3 BSD Unix implementation of DARPA's TCP/IP [7] protocol devotes only 4 kilobytes (by default) to protocol buffer space [8], and such a small buffer can easily overflow. Throwing memory at the problem wastes resources and may not be possible in small systems found, for example, on factory floors.

1.3. Previous Work

Although many distributed systems employ broadcast and broadcast-based multicast, researchers have lent little attention to "backoff" algorithms. Gusella and Zatti [9] employed backoff to reduce Ethernet collisions. Recipients held their responses for an additional random delay drawn from the uniform distribution on some interval. Our problems differ fundamentally. We minimize the expected time to collect all responses to a multicast, explicitly introducing the number of times the sender may retransmit the multicast. The system designer chooses the number of broadcast-based retransmissions that he will endure (Additional broadcasts extraneously interrupt those recipients that have

successfully responded). Knowing this, the number of buffers available, and the sender's service time distribution, we minimize the expected number of buffer overflows by deriving each round's optimal timeout and recipients' optimal common arrival time distribution. Recipients choose their backoff functions such that the combination of their natural response time distribution and backoff time function is distributed with this optimal distribution. We assume that the sender includes a bit-map of successfully acknowledged sites with each retransmission. This processes can be approximated and implemented in real systems.

1.4. Outline of The Paper

In Section 2, we consider overflow of single buffer systems. This illustrate the techniques we use to analyze overflow of multiple buffer systems and calculate the expected number of buffer overflows (back-to-back message loss) of single buffer systems. We repeat these analyses for several response time and buffer service time distributions. In Section 3 we derive a dynamic program that calculates the expected number of overflow losses for multiple buffer systems with general response time and exponential buffer service time distributions and for exponential response time and deterministic service time distributions. In Section 4, we pose the problem of finding the optimal timeout, common arrival distribution and backoff function that minimizes the expected number of buffer overflows during a round of multicast. For two recipients, exponential buffer service time, and a one buffer system, we derive the exact optimal response time distribution that minimizes the expected number of losses for a given round timeout. We apply these concepts in several examples. Finally, in Section 5 we review our findings and outline our future research plans.

2. Buffer Overflow of Single Buffer Systems

The number of buffer overflows depends upon the recipient response time and buffer service time distributions. We restrict our analysis to single buffer systems in this section because they are easier to analyze than multiple buffer systems, and because we can express their expected number of buffer overflows in closed form. We begin with the analysis of systems with preassigned response instants. After explaining why preassigned response instants are impractical, we devote the rest of the section to analyzing multicast systems with independent, identically distributed (*i.i.d.*) response time distributions. Single buffer systems suffer overflow if the interarrival time between subsequent responses, the back-to-back inter-message time, is less than the buffer service time. Borrowing the notation of queueing theory, we denote a multicast to N recipients, which respond with common response time distribution R, to a K-buffer server with service time distribution S, as an $R^N/S/1/K$ system.

We defer the analysis of exponential response time, exponential buffer service time distributions to Section 3, the discussion of multiple buffers systems, and start by analyzing constant buffer service time systems. Constant service time models network interface behavior well. The time to move a message from the network interface to the operating system is the sum of the interrupt service latency and the time for the DMA transfer between interface and memory. If the responses are of equal size, this time is roughly constant. (Some interfaces do not buffer entire messages, buffering a few bytes at a time in a FIFO instead.)

2.1. Preassigned Response Instants, $D_N/M/1/1$

In theory, we could assign a response instant y_i to each recipient. For example, we could assign to each recipient i, a unique instant $i\delta$ at which it should respond to the multicast sender. This system is easily modeled as its interarrival times are constant. If the buffer service time is exponentially distributed with mean $1/\mu$, then the expected number of overflows is the expected number of buffer service time intervals longer than δ. Using the method of indicator variables introduced below, we can show that

$$E[\,Losses\,] = \sum_{j=2}^{N} e^{-\delta\mu} = (N-1)\,e^{-\delta\mu}.$$

If the buffer service time is a constant β, no overflow occurs when $\delta > \beta$. Although preassigning response instants exhibits few losses, it requires maintaining a group list, requires accurate deadline scheduling, assumes that all recipients can calculate their responses before their mandated response instant expires, and can be unnecessarily slow when we do not require that all recipients respond. To illustrate this last point, consider the problem of finding a lightly-loaded site for balancing. Theimer [10] suggests that sites schedule their responses proportionally to their loads and not respond at all if they are unwilling to accept more jobs. Assigned response instants increase the expected time to receive a small number of responses, and is too rigid of a policy. We will consider *i.i.d.* random, response time distributions for the remainder of this section.

2.2. Upper Bound for $M^N/D/1/1$ Systems

Each recipient i responds at instant y_i independently of other recipients, where y_i is drawn from the exponential distribution with mean $1/\lambda$,

$$Pr(\,y_i \le t\,) = F(t) = 1 - e^{-\lambda t}.$$

We employ order statistics [11] to calculate the expected number of buffer overflows.

Briefly, the order statistics $y_{(1)}, y_{(2)}, \ldots, y_{(N)}$ of N *i.i.d.* random variables $y_1, y_2, \ldots, y_N$ are the y_i's sorted in increasing order. That is

$$y_{(1)} < y_{(2)} < \cdots < y_{(N)}.$$

The joint distribution of the order statistics of N *i.i.d.* continuous random variables is

$$n \prod_{i=1}^{N} f(y_i).$$

If $1/N\lambda$, the initial expected interresponse time, is not significantly larger than the constant buffer service time β, responses arrive closely spaced and easily overflow the buffer. The expected time to receive all N responses is the expected value of the slowest of the N independent response instants,

$$E[y_{(N)}] = \lambda^{-1}\,(1+2^{-1}+\ldots+N^{-1}),$$

the maximum of N *i.i.d.*, exponentially distributed random variables.

Recall that the sum of exponentially distributed random variables is itself exponentially distributed with rate equal to the sum of the individual rates. Immediately after the multicast, but before any recipient responds, the responses arrive at rate $N\lambda$. After the first response, this drops to $(N-1)\lambda$, and each

subsequent response decreases the future arrival rate by λ.

Since responses require buffer service time β, the buffer overflows if any interarrival interval is shorter than β. The probability Φ that no two responses arrive within time β, that is the probability that no responses are lost, is the product of the probabilities that all $N-1$ interrarrival times exceed β. Since $y_{(i+1)}$ arrives with rate $(N-i)\lambda$, the probability that the interval between $y_{(i)}$ and $y_{(i+1)}$ exceeds the service time β is

$$Pr[y_{i+1} - y_i > \beta] = e^{-(N-i)\lambda\beta}. \quad (1)$$

The probability Φ then is the product of the probabilities that all interarrival intervals exceed β,

$$\Phi = \prod_{j=1}^{N-1} e^{-\beta j \lambda} = e^{-\beta\lambda \sum_{j=1}^{N-1} j} = e^{-\beta\lambda N(N-1)/2}.$$

We discover that the mean response time $1/\lambda$ must grow quadratically with N to achieve a small probability of overflow.

The expected number of interarrival intervals shorter than β bounds the expected number of buffer overflows. We employ the method of indicator variables to derive this upper bound, which is a more convenient expression than the exact number of overflows calculated later in this section.

Let the indicator variable $I_i(\beta)$ be

$$I_i(\beta) = \begin{cases} 1 & \text{if } (y_{(i+1)} - y_{(i)}) \le \beta, \\ 0 & \textit{otherwise}. \end{cases} \quad (2)$$

The expected number of short intervals is

$$E[\sum_{i=1}^{N-1} I_i(\beta)] = \sum_{i=1}^{N-1} E[I_i(\beta)] = \sum_{i=1}^{N-1} Pr[y_{(i+1)} - y_{(i)} \le \beta].$$

The probability that $(y_{(i+1)} - y_{(i)}) \le \beta$ is given by (1). The expected number of short intervals bounds the expected number of overflow losses, and grows tighter with decreasing expected number of overflows. Keep in mind that the exponential distribution's weight extends to infinity, an infinite horizon, and additional losses are possible for responses that arrive after the round's timeout expires. The expected number of losses is bounded above by

$$E[Losses] \le E[\sum_{i=1}^{n-1} I_i(\beta)] = \sum_{j=1}^{N-1} (1 - e^{-j\lambda\beta})$$

$$\le N - 1 - \frac{e^{-\beta\lambda} - e^{-\beta N\lambda}}{1 - e^{-\beta\lambda}}. \quad (3)$$

2.3. Upper Bound for $U^N/D/1/1$ Systems

In this case, each recipient i responds at instant y_i drawn independently from the uniform distribution on the interval $(0, \tau)$. The expected time to receive all N responses is the expected value of the N^{th} order statistic $y_{(N)}$,

$$\frac{N\tau}{N+1}.$$

The probability density function of the order statistics of N uniformly distributed random variables is

$$f(y_{(1)}, y_{(2)}, \ldots, y_{(N)}) = \frac{N!}{\tau^N}.$$

The probability of no buffer overflow, Φ, equals the probability that all $N-1$ interarrival intervals exceed β, and can be found by evaluating the integral[1]

$$\Phi = \frac{N!}{\tau^N} \int_0^{\tau-N\beta} \int_{y_1+\beta}^{\tau-(N-1)\beta} \cdots \int_{y_{N-1}+\beta}^{\tau} dy_N \cdots dy_1.$$

The first integral corresponds to the arrival time of the first response. The lower limit of each subsequent integral is β greater than its predecessor. This represents the constraint that each interarrival period exceed β. Integrating yields

$$\Phi = \sum_{i=0}^{N} \binom{N}{i} (\frac{\beta}{\tau})^i (N-1)^i (-1)^i = \left[1 - (N-1)\frac{\beta}{\tau}\right]^N. \quad (4)$$

We have calculated the probability Φ of no overflow loss given the response time is uniformly distributed on $(0, \tau)$. Employing indicator variable $I_i(\beta)$, defined in (2), we can bound the expected number of responses lost to buffer overflow by the expected number of interarrival intervals shorter than β.

$$E[\sum_{i=1}^{n-1} I_i(\beta)] = \sum_{i=1}^{n-1} E[I_i(\beta)] = \sum_{i=1}^{n-1} Pr[y_{(i+1)} - y_{(i)} \le \beta]. \quad (5)$$

Each term in summation (5) corresponds to an integral similar to the one shown below, and it turns out that these integrals are identical. The integral corresponding to the summation's last term is

$$1 - \frac{N!}{\tau^n} \int_0^{\tau-\beta} \int_{y_1}^{\tau-\beta} \cdots \int_{y_{n-2}}^{\tau-\beta} \int_{y_{n-1}+\beta}^{\tau} dy_N \cdots dy_1. \quad (6)$$

Substituting (6) into (5) and adding up terms, we find the expected number of closely spaced responses, an upper bound on the number of losses, is

$$E[losses] \le (N-1)(1 - \sum_{i=0}^{N} \binom{N}{i} (-1)^i (\frac{\beta}{\tau})^i)$$

$$\le (N-1)(1 - (1 - \frac{\beta}{\tau})^N). \quad (7)$$

We employed order statistics to arrive at the probabilities the interrarrival interval exceeds β. However there exists an elegant proof which yields the same. Place $n+1$ points uniformly on the circle with circumference τ. Cut the circle at one of these points and label one end 0 and the other τ. From symmetry arguments, one arrives at (7).

3. Overflowing Multiple Buffers

We begin this section multiple buffer systems by finding the expected number of overflows when both the recipient response time and the sender's buffer service time are distributed exponentially, $M^N/M/1/b$. Next we develop a dynamic program that calculates overflow losses for a general *i.i.d.* response time distribution with an exponentially distributed buffer service time, $G^N/M/1/b$. Finally, we apply this program when the recipient's response time distribution is uniform, $U^N/M/1/b$.

Figure 3 plots the number of overflows for various numbers of recipients, numbers of buffers, and response time

[1] This problem appears in volume 2 of [12] as problem 24, Section I.13.

distributions of equal expected completion time $y_{(N)}$. Notice how the response time distribution significantly affects the expected number of losses. This effect increases with the number of buffers. We devote this section to deriving expressions that accurately predict this behavior.

3.1. Preassigned Response Instants, $D_N/M/1/b$

Assign to each recipient a unique time $k\delta$ at which it schedules its response. We calculate the expected number of losses by observing that a loss occurs if all b buffers are occupied when a recipient responds. Define $L_n(s)$ as the expected number of losses when n recipients respond, s buffers begin occupied, and the first of the n responses arrives immediately.

$$L_n(s) = \sum_{i=0}^{s+1} L_{n-1}(s+1-i)\, P_{s+1,i}$$

$$L_n(b) = 1 + \sum_{i=0}^{b} L_{n-1}(b-i)\, P_{b,i}$$

$$P_{i,j} = (\mu\delta)^i\, e^{-\delta\mu}/i!$$

$$P_{i,i} = 1 - \sum_{j=0}^{i-1} P_{i,j}$$

In these equations, we sum over i, the number of buffers that are emptied during time δ before the next response arrives. When the next arrival occurs, $s+1-i$ buffers are full, 1 for this arrival, $s-i$ for the previously buffered arrivals that were not served in the interim. The probability that i, $i < s+1$, of the full buffers are emptied is simply the probability that i Poisson-distributed events occur during time δ. The probability that all $s+1$ are emptied is the probability that non of the events $i \le s$ occur.

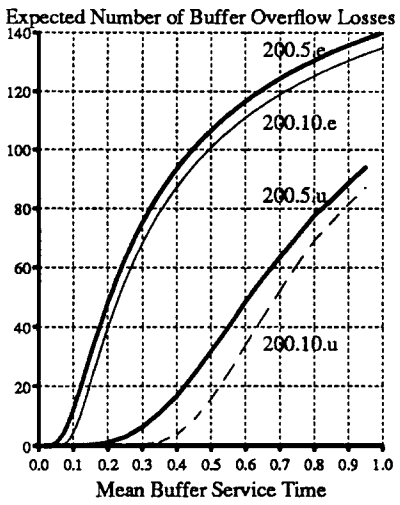

Figure 3. Multiple Buffer Overflow for (N,b)={(200,5),(200,10) }. $M^N/M/1/b$ and $U^N/M/1/b$ systems.

3.2. $M^N/M/1/b$ Systems

The problem of finding the number of buffer overflows given both exponential response and service time distributions is relatively easy and can be solved in numerous ways. We begin by drawing a two-dimensional Markov chain, where state (i,j) means i responses are outstanding and j buffers are full (see Figure 4). When $b=1$, we can write the expression for the expected number of buffer overflows by inspection.

$$E[losses] = \sum_{j=1}^{N-1} \frac{j\lambda}{\mu+j\lambda}.$$

Alternatively, we can write a fast dynamic program [13] to count the expected number of buffer overflows.

$$L_n(0) = L_{n-1}(1) \tag{8}$$

$$L_n(s) = P_{n,s} L_{N-1}(s+1) + (1-P_{n,s}) L_N(s-1)$$

$$L_n(b) = P_{n,b}(1 + L_{N-1}(b)) + (1-P_{n,s}) L_N(b-1)$$

$$P_{n,b} = \frac{n\lambda}{n\lambda + u_0(s)\mu}$$

where u_0 is the unit step function.

This program can be quickly calculated bottom up. In the next section we present a brute force technique that yields an exact solution for general response time distributions.

3.3. $G^N/M/1/b$ Systems

The dynamic program below finds the number of buffer overflows suffered by multiple buffer, general response time, exponential service time systems. The calculation is reminiscent of $G/M/1/b$ loss system analysis of classical queueing theory. Let $L_n(s,t)$ be the expected number of overflows given that s of the b buffers start full, the last arrival occurred at time t, and n additional arrivals remain outstanding. $L_n(s,t)$ is an integral over the arrival time, x, of the next arrival. Since all arrivals are drawn from a common response time distribution, we can easily express the probability density function, f_n, of the next arrival.

The probability distribution function of each of the remaining n responses, conditioned on the arrival time t of the last arrival, $F(y \mid y > t)$, is

$$Pr(y \le x \mid y > t) = \frac{P(y \le x)}{P(y > t)} = \frac{F(x)}{1-F(t)}.$$

Similarly,

$$Pr(y > x \mid y > t) = \frac{P(y > x)}{P(y > t)} = \frac{1-F(x)}{1-F(t)}.$$

The probability density function of the next arrival is then the product of this density and the probability that the remaining $n-1$ responses arrive later, all multiplied by n, the number of ways to select the first arrival.

$$f_n(x) = n \left[\frac{1-F(x)}{1-F(t)}\right]^{n-1} \frac{f(x)}{1-F(t)}$$

We now unveil the recursive expression for the expected number of overflows, $L_n(s,t)$.

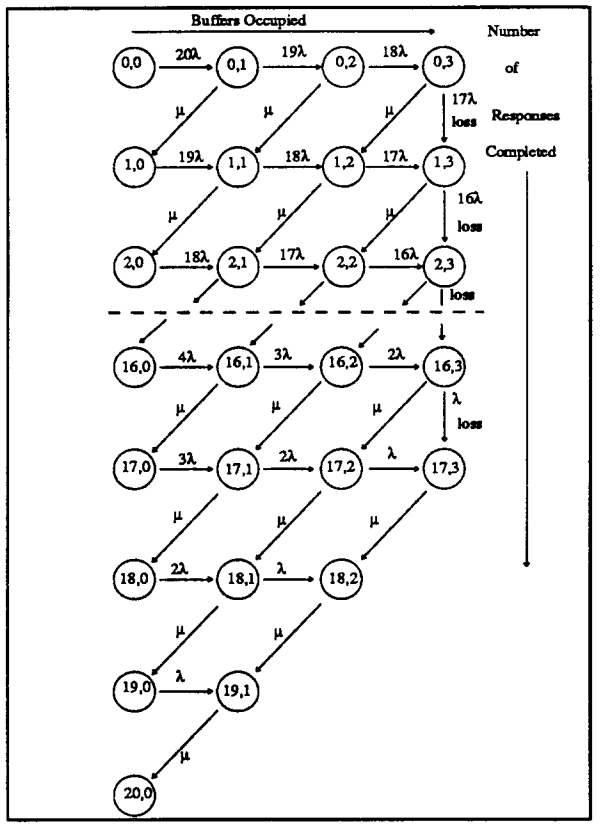

Figure 4. Markov chain that corresponds to the $M^{20}/M/1/3$ system.

$$L_n(s,t) = \int_t^{\infty} f_n(x) \sum_{i=0}^{s} L_{n-1}(s+1-i,x)\, p_{s+1,i}(x-t)\, dx. \tag{9}$$

$$L_n(b,t) = \int_t^{\infty} \left[f_n(x) \sum_{i=1}^{b} \left[L_{n-1}(b+1-i,x)\, p_{b,i}(x-t) \right] + (L_{n-1}(b,x)+1)\, p_{b,0} \right] dx$$

$$L_0(s,t) = 0.$$

$$p_{s,i}(x) = \begin{cases} (\mu x)^i e^{-\mu x}/i! & i<s, \\ 1 - \sum_{k=0}^{s-1} p_{s,k}(x) & i=s. \end{cases}$$

The sums are over the number of buffers serviced before the next response arrives at time x. We sum the expected number of overflows that occur when the remaining $n-1$ responses arrive given the $s+1-i$ buffers are occupied. The number $s+1-i$ proceeds from the s buffers previously occupied, one more for the new arrival, less the i buffers that were served during time x that no responses arrived.

When all b buffers are full, then a loss occurs with probability $p_{b,0}$, the probability that no buffers are emptied before the next response arrives. Although formidable in appearance, we can integrate expression (9) in polynomial time. For example, we now apply (9) to the case of uniformly distributed response times.

3.4. $U^N/M/1/b$ Systems

We can more easily calculate the solution to dynamic program (9) for uniformly distributed response times than for any other response time distribution. Exploiting the fact that the arrival distribution of the remaining responses is also uniform when it is conditioned on knowing the arrival instant of the most recent response, we redefine $L_n(s,t)$ to be the expected number of losses when n responses remain outstanding given that s buffers start full and the remaining responses arrive uniformly on $(0,t)$.

$$L_n(s,t) = \frac{n}{t} \int_0^t \left[\left[\frac{t-x}{t} \right]^{n-1} \sum_{i=0}^{s} L_{n-1}(s-i+1,t-x)\, p_{s,i}(x) \right] dx \tag{10}$$

$$L_n(b,t) = \frac{n}{t} \int_0^t \left[\left[\frac{t-x}{t} \right]^{n-1} \sum_{i=1}^{b} L_{n-1}(b-i+1,t-x)\, p_{s,i}(x) + (1 + L_{n-1}(b,t-x)\, p_{b,0}(x)) \right] dx$$

For single buffer systems with uniform responses on $(0, \tau)$, this dynamic programs yields the closed form solution

$$n!(n-1)\left\{ (\mu\tau)^{-1} \sum_{i=0}^{n} (\mu\tau)^{-i} (-1)^i/(n-1-i)! + (-1)^n e^{-\mu\tau} (\mu\tau)^{-n} \right\}.$$

Although closed form solutions for arbitrary numbers of buffers ($b \neq 1, b \neq N-1$) are not possible, we can always integrate (10). For example, expected overflow losses for $N=4$ recipients that arrive on (0, 1) and with a b-buffer system are given by

$b=1$: $12((\mu^{-1}-3\mu^{-2}+6\mu^{-3}-6\mu^{-4}) + e^{-\mu}(6\mu^{-4}))$.

$b=2$: $12((2\mu^{-2}-6\mu^{-3}+4\mu^{-4}) + e^{-\mu}((3\mu)^{-1}+2\mu^{-2}+2\mu^{-3}-4\mu^{-4}))$.

$b=3$: $12((2\mu^{-3}-6\mu^{-4}) + e^{-\mu}(\mu^{-2}+4\mu^{-3}+6\mu^{-4}))$.

Although the uniform response time distribution suffers fewer expected losses than the exponential distribution, it does not minimize buffer overflow. Finding the optimal, *i.i.d.* response time distribution that minimizes the number of buffer overflows

depends on the service distribution, the number of buffers, the number of responses, and the number of buffers that start full. We consider this problem in Section 4.

4. Finding the Backoff Algorithm and Round Timeouts

In many situations, all the recipients calculate and transmit their responses to the sender within a brief time of each other, and the sender loses some or many of these responses, and must request that certain sites repeat their responses. Demanding that each recipient delay its response for some random time alleviates the buffer overflow. We call this delay recipient *backoff*, and, in this section, construct an optimal backoff algorithm based on each recipient's measured response time distribution. Our algorithm suffers the fewest possible expected number of overflows, given that all recipients must respond before the round's timeout expires. We employ the backoff algorithm to minimize our multicast's cost metric, a linear, weighted sum of the number of broadcasts, number of responses, and overall elapsed time (latency). Alternatively, the sender can choose the number of broadcast-based rounds beforehand, and calculate each round's timeout immediately prior to broadcasting the round, so as to minimize the overall latency.

Finding the *i.i.d.* arrival time distribution that minimizes buffer overflow, given the number of recipients, number of buffers, round timeout, and buffer service distribution, we believe, is an unsolved problem. We call this the *Dynasty Problem*[2] and begin by solving it for two recipients, one buffer, and exponential buffer service time distribution. We note that its solution is composed of a uniform and a bimodal distribution, and our simulation study suggests that such a distribution is optimal for any number of recipients and buffers.

4.1. Optimal Response Time Distribution, Two Responses, One Buffer

Consider the problem of finding the common *i.i.d.* response time density $h(y)$, $0 \le y \le 1$, that minimizes the number of overflows of a single buffer server when there are two responses, and buffer service time is exponentially distributed with mean $1/\mu$. Or equivalently, find the common *i.i.d* response time density that minimizes the probability that the first response is still in service when the second arrives. If it is, the second response overflows the buffer, otherwise it does not.

The expected number of overflows, or this probability, is

$$2\int_0^1\int_{y_1}^1 e^{-(y_2-y_1)\mu}\, h(y_2)\, h(y_1)\, dy_2\, dy_1 \qquad (11)$$

constrained by the equations that make $h(y)$ a probability density function

$$2\int_0^1\int_{y_1}^1 h(y_2)\, h(y_1)\, dy_1\, dy_2 = 1\,,$$

and

$$h(y) \ge 0\,,\quad 0 \le y \le 1\,.$$

Although this appears solvable by the calculus of variations [14], the optimal distribution $h(y)$ does not have continuous first and second derivatives, a requirement for that technique. Instead, we transform (11) into a discrete optimization problem by subdividing (0, 1) into M identical subintervals, and apply the method of Lagrange multipliers [15] to find the optimal, discrete distribution.

Denote the number of losses by $L(\mathbf{p})$.

$$L(\mathbf{p}) = \sum_{i=1}^{M} p_i{}^2 + 2\sum_{i=1}^{M}\sum_{j=i+1}^{M} p_i\, p_j\, e^{-(j-i)\mu/M}\,.$$

We introduce Lagrange multipliers to incorporate the constraint equations.

$$l(\mathbf{p},\lambda) = L(\mathbf{p}) + \lambda\,((\sum_{i=1}^{M} p_i) - 1\,).$$

$$\nabla_{\mathbf{p}}\, l(\mathbf{p},\lambda) = \mathbf{0}\,.$$

$$\nabla_{\lambda}\, l(\mathbf{p},\lambda) = 0\,.$$

We solve this system of $M+1$ linear equations for the discrete solution $\mathbf{p}$. The endpoint probabilities p_1 and p_M are equal, and the interior-point probabilities are equal.

$$p_1 = p_M = \frac{e^{\mu/N}}{N(e^{\mu/N}-1)+2},$$

$$p_2 = \cdots = p_{M-1} = \frac{e^{\mu/M}-1}{M(e^{\mu/M}-1)+2}.$$

Taking the limit as integer M becomes large and substituting $(1+\mu/M)$ for $e^{\mu/M}$, we find the weight at the interval's endpoints is conserved

$$p_1 = p_M = \frac{1}{\mu+2}\,,$$

and the probability density of the interior points remains uniform. The optimal *i.i.d* continuous density function is

$$h(y) = \frac{\delta(y) + \mu + \delta(1-y)}{\mu+2}, \qquad (12)$$

where $\delta(y)$ is the Dirac delta function.

We see $h(y)$ is the superposition of a uniform distribution and two impulses, one at either endpoint. As the mean service time $1/\mu$ decreases, the optimal distribution approaches the uniform distribution. As the mean service time increases, the optimal distribution approaches the bimodal distribution with equal weight of one half at the interval's endpoints 0 and 1.

We find the expected number of losses by carrying out integral (11), substituting (12) for $h(y)$,

$$E[losses] = \frac{2}{2+\mu}\,.$$

We plot the expected number of losses for the optimal distribution (12), the uniform distribution, and the bimodal distribution with probability one half at each end point in Figure 5.

The bimodal arrival distribution always outperforms the uniform arrival distribution [17], showing more improvement as the number of recipients approaches the number of buffers. Assume we knew the optimal arrival distribution on an interval $(0, \tau)$. If we found a backoff algorithm that mapped each site's measured response time distribution into the optimal response time distribution, then we would minimize buffer overflow.

[2] The name *Dynasty* recognizes a Berkeley restaurant, dubiously honored with the city's worst health inspection record.

[3] A different problem is solved in [16] using similar methods.

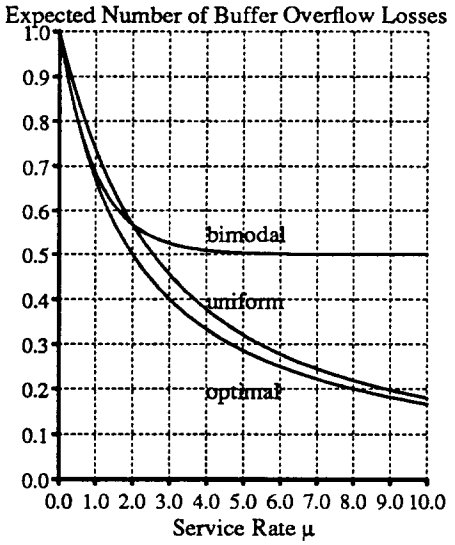

Figure 5. Losses versus response time distribution for 2 responses, exponential buffer service, and one buffer.

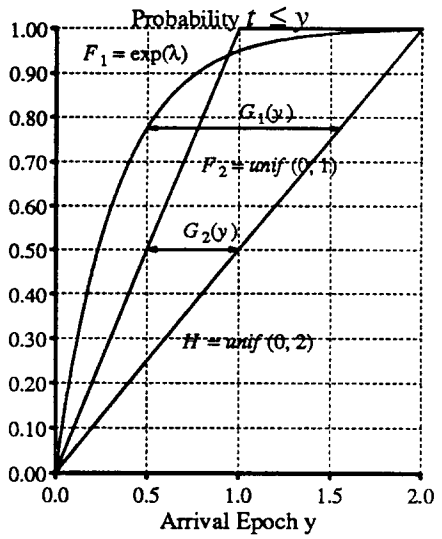

Figure 6. Several backoff functions G_i.

4.2. Our Backoff Algorithm

Since the recipients' response time distributions are neither identical to one another, nor equal to some well characterized distribution, they must be individually measured. Denote recipient i's measured response time distribution by F_i and its site-dependent backoff algorithm by G_i. Denote recipient i's observed response time to the multicast by y_i, and, finally, assume the optimal arrival distribution that minimizes buffer overflow given N recipients, round timeout τ, and b buffers is known, and denoted by $H(N, b, \tau)$. In this section we construct an optimal, backoff function $G_i = G_i(F_i, \tau, y_i)$ that minimizes the expected number of buffer overflows. In the next section we show how to select the round's timeout τ.

We want a backoff function G_i such that, when added to the site's response time y_i, it transforms the site's response time distribution F_i into the optimal arrival distribution H. If y_i exceeds τ then the response y_i becomes ready during some subsequent round of multicast, and we delay it no further. In real systems late responses may be processed; here, we consider them lost. We construct our site specific backoff function G_i as follows. It is the time difference between the optimal distribution and the response time distribution corresponding to the observed response time. We defer considering response time distributions F_i that lie beneath H (contrast Figure 6 and Figure 8) until later in the section.

$$G_i = H^{-1}(F_i(y_i)) - y_i.$$

Suppose the optimal arrival distribution were uniform and the service time distributions F_i lie above it. The equation below would give the site specific backoff functions.

$$G_i(y_i) = max\left(\frac{y_i F_i(y_i)}{F(\tau)}, 0\right). \qquad (13)$$

In Figure 6 we plot probability distribution functions H (supposed uniform), F_1 (an exponential), and F_2 (another uniform), and the backoff functions for site one and two, G_1 and G_2.

We want to map F_1, exponentially distributed with parameter λ, onto the uniform distribution on $(0, \tau)$. Since the exponential distribution's tail is infinitely long, some responses may arrive after the round's timeout τ expires. Assume such responses are lost. However, every response that arrives before τ appears to arrive uniformly on $(0, \tau)$. Applying equation (13), the backoff function is

$$G_1(y_1) = max\left(\frac{\tau(1 - e^{-\lambda y_1})}{1 - e^{-\lambda\tau}} - y_1, 0\right).$$

We want to map F_2, uniformly distributed on $(0, b)$, onto the uniform distribution on $(0, \tau)$. G_2 depends on whether the b exceeds or falls short of τ. If b exceeds τ, we assume responses that arrive after the round's timeout τ expires are lost, and we do not further delay them.

$$G_2(y_2) = \begin{cases} 0 & \text{if } b > \tau, \\ y_2(\tau - b)/b & \textit{otherwise.} \end{cases}$$

If some third site exhibited an instantaneous response time, F_3, then the backoff function would simply be a uniformly distributed random variable.

$$G_3(y_3) = \textit{uniform}(0, \tau).$$

This foreshadows the section on discontinuous response time distributions. If any F_i contains a discontinuous jump, then G_i maps this jump onto a uniformly distributed random backoff time of an appropriate duration.

4.3. Selecting a Round's Timeout

The problem of selecting a round's timeout is related to the problem of detecting failed sites, mentioned earlier. Recall that the sender determines that a site has failed if it fails to respond to several transmissions. Existing protocols attempt 4 or 5 retransmissions with timeouts either determined by binary-exponential backoff or simply set to a few seconds, or tens of seconds. If a recipient fails no respond after all of these attempts pass, the sender assumes it has failed. On one hand, if we set the

timeouts too short, each round results in many buffer overflows, and 5 rounds may be insufficient to determine site failure. On the other hand, if we set the timeouts too long, we needlessly increase the multicast's latency. Selecting timeouts poses difficult optimization problems, which we illustrate with several examples. Let's assume that recipients respond uniformly on a prescribed interval.

Example. $U^{20}/D/1/1$, Two Rounds.

Consider a two-round multicast. We want to minimize the latency, $\tau_1 + \tau_2$, such that after the second round, the probability that one or more responses remain outstanding due to buffer overflow is less than ε. We permit the second round timeout to depend on the number of overflows experienced during the first round. We must solve the optimization problem

$$minimize \quad E\left[\tau_1 + \tau_2\right] \tag{14}$$

such that

$$\Phi(k, \tau_2(k)) > 1 - \varepsilon, k = 0...n-1.$$

where Φ is given by (4).

In essence, we must select the optimal value for τ_1 such that the sum of τ_1 and $E[\tau_2]$ is minimized. For this we need the discrete probability distribution of losing k responses, P_k, which is in general difficult to calculate. As this example is no exception, we choose to approximate this system's distribution by the binomial distribution. (In [17] we employ a better, albeit more sophisticated approximation).

The binomial approximation says that a response overflows with probability that is independent of all other responses. We made a similar approximation in Section 2.3 where we bounded the expected number of losses that this system experiences. The probability p that a given responses overflows the sender's buffer is bounded by the second term of ((7)).

$$p = (1 - \frac{\beta}{\tau})^N .$$

Since only $N-1$ of the possible N responses can overflow, the binomial approximation to the desired P_k is

$$P_k(N-1, \tau_1, 1) = \binom{N-1}{k} p^{N-1-k} (1-p)^k .$$

In Figure 7 we constrast this approximation with the distribution obtained through simulation. It lies to the right of the value obtained by simulation because (7) is an upper bound on the number of losses.

If the sender loses k responses during the first round, then it must choose τ_2 such that the probability Φ that it collects the remaining k responses during the second round exceeds $1 - \varepsilon$. We invert (4) and obtain

$$\tau_2(k) = \frac{1 - (1-\varepsilon)^{1/k}}{k - 1}$$

We can now calculate $E[\tau_2]$.

$$E[\tau_2] = \sum_{k=0}^{n-1} P_k \, \tau_2(k).$$

Applying this to the $U^{20}/D/1/1$ system, we examine how the choice of the first round timeout τ_1 affects the expected value

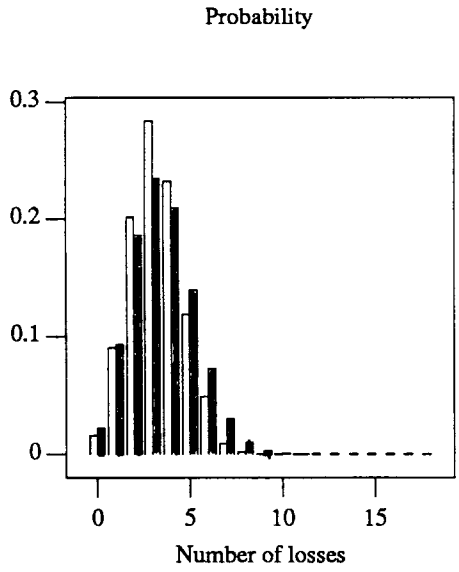

Figure 7. Probability distribution, $U^{20}/d/1/1$, τ=1, service time β=.01. The approximation is shaded.

of the total latency, $E[\tau_1 + \tau_2]$. In Figure 8 we plot total latency versus the choice of τ_1. When τ_1 is much smaller than optimal, the expected latency approaches the latency of a one round multicast constrained by the same condition that $\Phi > 1 - \varepsilon$. When τ_1 is much larger than optimal, the expected latency approaches τ_1.

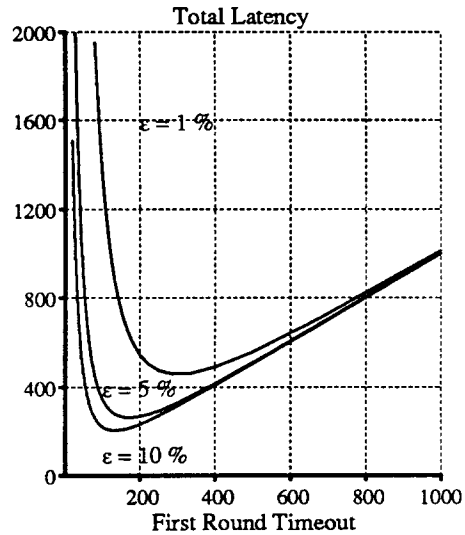

Figure 8. Total latency of a two round multicast as a function of the first round timeout τ_1. $U^{20}/D/1/1$, Service time $\beta = 1$. One round multicasts require τ of 38,000 (ε = 1%), 4,600 (ε = 5%), and 3,600 (ε = 10%).

5. Conclusions

Existing methods to analyze finite buffer systems [2] do not apply to multicast; while the overflow analyses presented in this paper provide insight into making multicast fast. Single buffer systems treated in Section 2 yield simple expressions for the expected number and distribution of losses. Solving these systems leads to the techniques used to solve multiple buffer systems. Overflow of multiple buffers, treated in Section 3, is frequently a problem of multicast in distrubted systems. It has been reported in the literature as back-to-back packet losses of Ethernet interfaces, and can occur within the layers of software between the network interface and the user program. The backoff algorithm and the timeout selection problem presented in Section 4 minimize the latency of reliable multicast, and minimize buffer overflows regardless of whether they occur at the network interface or between user-system software boundaries [8] (recall Figure 1).

These results can help system designers optimize multicast protocols, decide on the number of buffers to devote to a multicast sender, decide the cut off between using hardware broadcast and unicast primitives, and choose the number of rounds of multicast. They place selection of the round timeout on firm mathematical ground. Beyond the local area network, these techniques apply overflow at network gateways caused by internet multicast [18] and overflow at LAN bridges caused by extended LAN multicast.

We have been able to eliminate several of the unrealistic assumptions mentioned in Section 1.1, to solve the Dynasty problem exactly for one buffer systems, and to calculate many of the overflow distributions not presented in this paper. In the furture, we intend to implement the backoff algorithm efficiently and determine how far from optimal a practical implementation must lie.

Acknowledgments

I wish to thank Domenico Ferrari for his support and confidence, and Carl Ponder and George Shanthikumar for several useful discussions. I am grateful to my office mates and friends Riccardo Gusella, Venkat Rangan, Mark Sullivan, Shin-Yuan Tzou, and Dinesh Verma for their comments on early versions of this paper and for suffering my exuberant interruptions.

References

1. D. Clark, "The Structuring of Systems Using Upcalls", *10th Symp. on Operating System Prin. 19*, 5 (Dec 1985), 171-180.
2. H. G. Perros and T. Altiok, "Approximate Analysis of Open Networks of Queues with Blocking: Tandem Configurations", *TSE 12* , 3 (March 86), 450-n461. North Carolina State University.
3. J. L. Wang and J. A. Silvester, "Throughput Optimization of the Adaptive Multi-Receiver Selective-Repeat ARQ Protocol Over Broadcast Links", *Proceedings of ICCC*, Tel Aviv, Israel, October 1988.
4. A. Ganz and I. Chlamtac, "Queueing Analysis of Finite Buffer Token Networks", *SIGMETRICS*, Santa Fe, New Mexico, May 1988, 30-36.
5. J. Chang and N. F. Maxemchuk, "Reliable Broadcast Protocols", *Trans. Computer Systems* 2, 3 (Aug. 1984), 251-273.
6. D. R. Cheriton and W. Zwaenepoel, "Distributed Process Groups in the V Kernel", *Trans. Computer Systems 3*, 2 (May 1985), 77-107.
7. E. J. Feinler, O. J. Jacobsen, M. K. Stahl and C. A. Ward, *DDN Protocol Handbood*, Defense Communications Agency, December 1985.
8. L. F. Cabrera, E. Hunter, M. J. Karels and D. A. Mosher, "User-Process Communication Performance In Netwoks of Computers", *TSE 14*, 1 (Jan. 1988), 38-53.
9. R. Gusella and S. Zatti, "An Election Algorithm for a Distributed Clock Synchronization Program", *IEEE 6th International Conference on Distributed Computing Systems*, Boston, May 1986.
10. M. M. Theimer and K. A. Lantz, Finding Idle Machines in a Workstation-based Distributed System, 8th International Conference on Distributed Computing Systems, June 1988.
11. S. M. Ross, *Stochastic Processes*, John Wiley & Sons, New York, NY, 1983.
12. W. Feller, *An Introduction to Probability Theory and Its Applications Volume 1 and 2*, John Wiley & Sons, New York, NY, 1970.
13. S. M. Ross, *Introduction to Stochastic Dynamic Programming*, Academic Press, New York, NY, 1983.
14. R. Weinstock, *Calculus of Variations*, McGraw-Hill, New York, NY, 1952.
15. M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice Hall, Englewood Cliffs, NJ, 1976.
16. B. Simons and L. G. Votta, "The Optimal Retry Distribution for Lightly Loaded Slotted Aloha Systems", *IEEE Transactions on Communications COM-33*, 7 (July 1985), 724-725.
17. P. B. Danzig, "Buffer Overflow in Multicast", *IEEE Trans. on Computers Special Issue on Performance Evaluation*, (Submitted).
18. S. Deering, "Multicast Routing in Internetworks and Extended LANSs", *SIGCOMM '88 Symposium*, Aug. 16-19, 1988, 55-64.